Dynamic Adaptive Streaming for Multi-Viewpoint Omnidirectional Videos

Xavier Corbillon
IMT Atlantique, IRISA, France
xavier.corbillon@imt-atlantique.fr

Francesca De Simone
DIS, Centrum Wiskunde & Informatica, The Netherlands
F.De.Simone@cwi.nl

Gwendal Simon
IMT Atlantique, IRISA, France
gwendal.simon@imt-atlantique.fr

Pascal Frossard
LTS4, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
pascal.frossard@epfl.ch

ABSTRACT

Full immersion inside a Virtual Reality (VR) scene requires six Degrees of Freedom (6DoF) applications where the user is allowed to perform translational and rotational movements within the virtual space. The implementation of 6DoF applications is however still an open question. In this paper we study a multi-viewpoint (MVP) 360-degree video streaming system, where a scene is simultaneously captured by multiple omnidirectional video cameras. The user can only switch positions to predefined viewpoints (VPs). We focus on the new challenges that are introduced by adaptive MVP 360-degree video streaming. We introduce several options for video encoding with existing technologies, such as High Efficiency Video Coding (HEVC) and for the implementation of VP switching. We model three video-segment download strategies for an adaptive streaming client into Mixed Integer Linear Programming (MILP) problems: an omniscient download scheduler; one where the client proactively downloads all VPs to guarantee fast VP switch; one where the client reacts to the user's navigation pattern. We recorded a one MVP 360-degree video with three VPs, implemented a mobile MVP 360-degree video player, and recorded the viewing patterns of multiple users navigating the content. We solved the adaptive streaming optimization problems on this video considering the collected navigation traces. The results emphasize the gains obtained by using tiles in terms of objective quality of the delivered content. They also emphasize the importance of performing further study on VP switching prediction to reduce the bandwidth consumption and to measure the impact of VP switching delay on the subjective Quality of Experience (QoE).

Part of the research published in this paper was conducted while Francesca De Simone was affiliated to the LTS4 at EPFL.
The authors thanks the Swisscom Innovation Lab for providing the omnidirectional cameras used to record the content and Meryem Wehbe who implemented the MVP 360-degree player used to collect the navigation patterns used in this study.

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06…$15.00
https://doi.org/10.1145/3204949.3204968

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; • **Computing methodologies** → **Virtual reality**;

KEYWORDS

Multi-viewpoint 360-degree Video, Omnidirectional Video, Discrete 6DoF, Viewport Adaptive Streaming

ACM Reference Format:
Xavier Corbillon, Francesca De Simone, Gwendal Simon, and Pascal Frossard. 2018. Dynamic Adaptive Streaming for Multi-Viewpoint Omnidirectional Videos. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3204949.3204968

1 INTRODUCTION

The current immersive multimedia services offering omnidirectional (also coined as 360°) video are typically designed to provide Virtual Reality (VR) experience with three Degrees of Freedom (3DoF). A user who watches a 360° video with a Head-Mounted Display (HMD) can choose the portion of the spherical content to view by rotating the head to a specific direction. Yet, to enable full immersion inside a VR scene, head rotation alone is not sufficient. The ability to perform translational movements inside the content is also required [8]. The user can then fully navigate the scene by virtually moving within it. VR applications allowing rotations and translations inside a virtual scene are referred to as six Degrees of Freedom (6DoF) applications. Yet the implementation of a continuous 6DoF VR application, i.e., *free-viewpoint*, without restriction in translations is still an open challenge [16].[1]

A first implementation of 6DoF VR applications consists in restricting the navigation to only predefined positions in space. We refer to a *multi-viewpoint (MVP) 360° video* system as a VR application where the scene is offered at some *viewpoints (VPs), i.e.* positions from which the scene can be viewed. To allow changes of VP, the same scene has to be simultaneously captured by multiple 360° cameras located at different positions in space.[2] *Google Earth VR* [1] is an example of such a MVP 6DoF VR application allowing the navigation into a virtual scene by changing the VP (see

[1] 6DoF VR will typically be addressed during the phase 2 of the Moving Picture Experts Group – Immersive (MPEG-I)
[2] Cameras can be either real cameras or virtual cameras based on view synthesis

Figure 1: Example of VP switching options appearing in a viewport in the *Google Earth VR* [1] interface, i.e., *walk-through* functionality. Each white sphere appearing in the viewport represents a VP that the user can switch to. The VP switch can be selected using a controller, whose virtual representation is appearing in the viewport as well.

Figure 1). However, it is to date restricted to static pictures instead of videos.

When a MVP omnidirectional video acquisition system is available, streaming MVP 360° videos to the end-user brings new challenges. We highlight in particular two challenges: (*i*) determining the service implementation design that enables a good immersion experience without explosion of the resource requirements in the delivery chain (server, network, client); (*ii*) designing the best data downloading strategy at the client side to maximize the Quality of Experience (QoE). Exploring these challenges and their possible solutions is the scope of this paper.

The first challenge is related to the implementation of the server. In the recent years, researchers have seek for solutions to stream *single-viewpoint* 360° videos. For obvious reasons of re-using existing delivery architectures, proposals aim to patch the technologies of dynamic adaptive streaming, typically MPEG Dynamic Adaptive Streaming over HTTP (DASH) [34, 36]: the server stores an *adaptation set*, *i.e.*, a set of multiple *representations* of the same content encoded at different bit-rates and resolutions. Each representation is divided in consecutive *segments* of fixed duration (ranging between 1 s and 5 s). To exploit the fact that, during the time course of an omnidirectional video segment, what is displayed in the HMD is only a fraction of the whole video (also denoted by *viewport*), the main emerging idea is to prepare video representations where the quality within the frame is not homogeneous [4, 14, 35]. Then, a client, who should select the representation to download for the next segment, takes into account not only a prediction of the bandwidth for the next seconds but also a prediction of the viewport position. Ideally, the viewport, which is delimited by a Field of View (FoV) covering a spherical portion of approximately 100°, both in vertical and horizontal direction, should have 4K resolution. By extension, the resolution of the 360° video corresponding to only one VP should be approximately 12K resolution [11]. By multiplying the number of VPs, the service provider should provision a large amount of resources to prepare, store, and deliver the data. The choices at the server side, which are both on the implemented technologies and the design of the application, are bound to the resource limitations on the whole delivery chain. Identifying the best design choices in this regard is thus an open research topic.

The second challenge is related to the selection of representations at the client side. Now, in addition to the two criteria that the client has to take into account to select the representation (bit-rate and head orientation), the translation movements in the 6DoF scene should be anticipated to enable fast VP switch. To the best of our knowledge, no previous work has studied *MVP* 360° video streaming. Nevertheless, it is reasonable to assume errors in translational movement predictions, which would affect the interactivity of the application and could significantly impact the user's QoE. The design of an accurate client adaptation strategy, including a 6DoF movement prediction module, is thus another open research topic.

In this paper, we analyze some MVP 360° adaptive streaming scenarios, we describe the general layout of the omnidirectional MVP acquisition configuration as well as the content navigation at user side. We define the adaptation strategy of the dynamic adaptive streaming client as an optimization problem and demonstrate that exploiting the knowledge upon the way the user is navigating the content can maximize the QoE, while minimizing the bandwidth consumption. More precisely, we propose four contributions in this paper:

- We discuss possible implementations of a MVP 360° adaptive streaming system from the server perspective, reviewing state of the art encoding solutions that could be used in this scenario.
- For a given adaptation set, we formulate the client adaptation logic, i.e., the algorithm that selects which representation to download for each segment, as a Mixed Integer Linear Programming (MILP) optimization problem to jointly (*i*) maximize the visual quality in the user's viewports, (*ii*) minimize the VP switching delay, and (*iii*) minimize the frequency and duration of video playout stalls, subject to bandwidth constraints. We consider the ideal scenario, where the client performs a perfect prediction of both his/her bandwidth as well as his/her navigation patterns, as an upper bound scenario for the optimization.
- We then study two extreme strategies for the realistic scenario in which the client does not predict the VP switch. The first strategy is a *proactive* algorithm where the client systematically downloads other VPs to anticipate VP switches and enable minimum switching delay. The second strategy is a *reactive* algorithm where the client downloads VPs only when the user commands a translational movement, such that VP switch requires some tolerance to delay. Both strategies bound the spectrum of all viewpoint-switching adaptation algorithms that the service providers could design.
- We acquired a four-minutes long 360° MVP video sequence and developed an interface for a 6DoF VR application to collect the navigation patterns of some users watching our video sequence. An analysis of the navigation patterns of our set of users is presented. We gathered multiple existing tools together to emulate the MVP 360° DASH streaming, to extract users' viewports from the reconstruct bit-stream with

non-homogeneous quality, and to compute some objective video quality metrics.

The paper is organized as follow. Section 2 presents the related work on the topic of 360° video streaming and multi-view adaptive video streaming. Section 3 describes the layout of a general MVP 360° content acquisition and navigation framework, introducing the notation and terminology used in the paper, and exhibits the options considered in this paper to encode a MVP 360° at the server side. Section 4 presents the client of a MVP 360° video streaming system, as well as the metrics used to model the user's QoE in such a system. Section 5 introduces the proposed formulation of the client adaptation logic as an optimization algorithm. Section 6 describes the MVP video content used for the evaluation, as well as the test conditions used to simulate the streaming session, and discusses the results. Finally, Section 7 concludes the paper.

2 RELATED WORK

We are not aware of any paper dealing with MVP omnidirectional video streaming for 6DoF VR. The bibliography that we report in the following is related to the streaming of single-camera 360° videos and to multi-view perspective (as opposed to omnidirectional) videos. We focus on those topics because MVP 360° videos can be seen as a mix between single-camera 360° videos and multi-view perspective videos and so innovation in one of those field may also be beneficial to MVP 360° video technology.

2.1 Single-Camera 360° Video

The number of studies related to 360° video streaming has exploded in the last couple of years. Considering the principles of viewport-adaptive video streaming as they have been explained in Section 1, these studies deal with (*i*) the encoding of quality-variable representations, (*ii*) the prediction of head orientation, and (*iii*) the mechanisms to request the right representations.

Quality-Variable Video Encoding. We distinguish two main approaches: one where the variation of the quality is done when the spherical video is projected into a plane [21, 22, 46], and one where the encoding of the projected frames enables differentiating the quality within the frame [6, 11]. We focus on the latter. The main approach that has been studied is related to the motion-constrained tile sets (MCTS) coding, which has been introduced with the High Efficiency Video Coding (HEVC) codec. This technique spatially splits a rectangular video into rectangular, independently decodable, non-overlapping regions [28]. Each tile can be extracted from the MCTS bit-stream into its own file, which can be independently downloaded. Encoding a video with the MCTS introduces a bit-rate overhead due to the tile headers and due to losses in compression efficiency [3]. On the other hand, tiling allows more flexible streaming strategies, especially in the context of omnidirectional videos where only a small portion of the content is displayed to the user at each instant in time. Typically, a planar omnidirectional video can be divided in tiles: for example, a *cube-map baseball layout*, depicted in Figure 2, can be divided in tiles where each cube face is a tile. The user can download in high quality only the tiles that are located in the portion of the sphere that is most likely to be attended. This approach, which is advocated in many studies [7, 10, 15, 30, 42], has been successfully implemented [7] and integrated into recent standards [14, 18, 29].

Figure 2: Example of omnidirectional video frame in *cube-map baseball layout*: the spherical frame is mapped to the plane via the cube-map projection and the faces of the cube are arranged into a rectangular frame with 3:2 aspect ratio by minimizing discontinuities between the cube faces.

User Behavior Prediction. As in traditional dynamic *rate-adaptive* streaming systems where the ability to predict the available bandwidth for the next seconds is a critical factor of implementation success [24, 25, 27], the ability to predict the head orientation of the user for the next seconds is a key element of *viewport-adaptive* streaming. Some datasets created by collecting data from users watching omnidirectional videos have recently been released with the objective to foster research activity in the area [5, 26, 40]. The known prediction algorithms, based on deep learning [9] or statistical approach [31, 32], do however not apply to MVP 6DoF video streaming applications, for which new datasets and new models will have to be developed.

Representation Request. To our knowledge, Zhou et al. [46] have published the only study that formally dissects existing viewport-adaptive streaming systems. They showed in particular that, in a popular 3DoF streaming system, clients start sending requests at time t for the segments that will be displayed at $t + 5$ s. During this time, it is possible that abrupt unpredicted head movements significantly impact the client download strategy. In extreme cases, the client can request a different representation *for the same segment*, which means that two representations are downloaded for the same segment to accommodate user behavior. Such double downloading, which is a waste of bandwidth, occurs as frequently as 32 % of the total number of segments. More generally, such waste of downloading is arguably the price to pay to deal with interactive and fuzzy behaviors. By downloading twice the same segment, the client can decide until the very last moment which segment to play to maximize user's QoE.

2.2 Multi-View Perspective Video

Multi-view video streaming considers content acquired by multiple cameras having limited disparity. Often, depth information is also available, which can be used to synthesise additional views. The studies addressing multi-view video streaming over HTTP can be clustered according to their focus on (*i*) the strategy used to create the adaptation set, thus, the multi-view video encoding strategy used to optimize the storage space at server side and allow rate-adaptive video transmissions and (*ii*) the optimal request or delivery scheme, to account for the priority of the view that clients request as well as the view-switching probabilities.

Multi-View Adaptation set. To exploit the redundancy that is present in multiple views of the same scene, Su et al. [37] propose an HEVC multi-view streaming system where the multi-view video and depth content is encoded using the scalable extension of HEVC. The scalable coding introduces dependencies between representations of different views. Scalable video coding of multi-view video including depth content is also used by Zhao et al. [45], who propose a cloud-assisted streaming system where virtual views can be synthesized at server or client side depending on the network conditions and the cost of the cloud-based server. Recently, Toni and Frossard [38] have proposed to formulate the adaptation set design at server side as an integer linear programming optimization problem, in order to optimize storage constraints while offering a good navigation quality to the different users.

Optimal Request or Delivery Scheme. Focusing on the adaptation logic at the client side, Hamza and Hefeeda [12, 13] propose a quality-aware rate adaptation method for free-viewpoint streaming, based on a rate-distortion model that relates the distortion of the texture and depth components of reference views and target virtual views. This model enables the client to find the best set of representations to request from the server. The adaptation strategy takes into consideration the user interaction with the scene, assuming that the navigation trajectory of the user is predicted at client side. The problem of minimising the latency of the view-switching at client side has been addressed by multiple works in literature. Xiao et al. [41] propose two view switching approaches which exploit data buffering at client side, in order to reduce the view switching delay. Carlsson et al. [2] propose a prefetching policy implemented at client side, so that the requested view and rate are adapted based on the stream switching probabilities and the current bandwidth constraints. Yun and Chung [43] use a buffer occupancy controller at client side, as well as parallel streaming and server push policy to minimize the view-switching delay. Finally, Zhang et al. [44] propose a priority-based adaptive scheduling algorithm implemented at the server side when multiple VPs are simultaneously transmitted over bandwidth constrained network to multiple clients.

In this paper, we study adaptive streaming for MVP 360° videos. To the best of our knowledge, neither the single-camera 360° video, nor the multi-view perspective video delivery are directly related to the system we address here. Regarding single-camera 360° video, we adopt most of the techniques that have been developed in the recent years to match the delivered content to the displayed viewport. But no previous work has dealt with multiple VPs. Regarding multi-view delivery, the cameras in our case exhibit a high variation of the disparity and only a fraction of a VP is displayed. Note however that multi-view techniques can be used to generate *virtual* cameras, which may become new VPs in the MVP 360° video system. But virtual VP synthesis is out of the scope of this paper.

3 MVP 360° FRAMEWORK AND NOTATION

In this section we introduce the geometry of a single VP, generalize the framework to a MVP system focusing on the VP switching conditions, and present the options available to a service provider to encode MVP 360° videos. We describe the general layout of content acquisition and navigation of MVP 360° content, introducing the notation and terminology used in the paper.

We consider the affine Euclidean space $\mathbb{R}^3$, with the orthonormal world reference frame $\left(O, \vec{\imath}_o, \vec{\jmath}_o, \vec{\mathbf{k}}_o\right)$, and the right hand rule to define rotations.

3.1 One Viewpoint

A fully omnidirectional camera can be considered as a calibrated central camera with center of projection $v \in \mathbb{R}^3$. The camera projects any point in $\mathbb{R}^3$ to a point on the spherical imaging surface of radius r, usually considered unitary, *i.e.*, the *viewing sphere* S_v centered at v. Formally, $S_v := \left\{p \in \mathbb{R}^3 : \|p - v\| = r\right\}$. The omnidirectional video signal is defined on S_v. The viewing sphere is associated to the orthonormal frame $\left(v, \vec{\imath}_o, \vec{\jmath}_o, \vec{\mathbf{k}}_o\right)$: a translation $T \in \mathbb{R}^3$ transforms $\left(O, \vec{\imath}_o, \vec{\jmath}_o, \vec{\mathbf{k}}_o\right)$ into $\left(v, \vec{\imath}_o, \vec{\jmath}_o, \vec{\mathbf{k}}_o\right)$.

A user watching the omnidirectional video is assumed to be at the center of the viewing sphere. At each instant in time, the user visualizes only a portion of the spherical surface, depending on his *viewing orientation*. During the video duration, the user can change his viewing orientation to navigate the video. We associate the user's head to the orthonormal frame $\left(v, \vec{\imath'}, \vec{\jmath'}, \vec{\mathbf{k}'}\right)$, where $\vec{\imath'}$ goes through the front of the user's head, $\vec{\jmath'}$ through his left ear, and $\vec{\mathbf{k}'}$ through the top of his head. A rotation $R \in \mathrm{SO}(\mathbb{R}^3)$ transforms $\left(v, \vec{\imath}_o, \vec{\jmath}_o, \vec{\mathbf{k}}_o\right)$ into $\left(v, \vec{\imath'}, \vec{\jmath'}, \vec{\mathbf{k}'}\right)$. Thus, at a given instant in time, the user's viewing orientation within the viewing sphere centered at v is uniquely identified by R. $\vec{\imath'} = R(\vec{\imath})$ is then the user viewing direction.

When the content is rendered to be visualized, for example via a HMD, the portion of the sphere surface corresponding to the user's viewing direction is projected to a planar segment tangent to it, called the *viewport*. A viewport is defined by the user's viewing direction, which identifies the point where the viewport is tangent to the sphere, its horizontal and vertical angular size, i.e. the FoV, and its pixel resolution. We consider the angular sizes and resolution fixed. Thus, the viewport depends only on the user's viewing direction and we can denote the viewport attended at an instant in time by the user within the viewing sphere centered at v by (v, R).

3.2 Multiple Viewpoints

An omnidirectional MVP video content corresponds to a finite set of L omnidirectional video sequences of the same scene, captured by omnidirectional cameras located at different positions in space, and synchronised in time, at frame precision. We refer to each omnidirectional video, corresponding to a camera at a particular position in space, as a *VP*. We denote by $\mathcal{V} = \{v_j\}$, with $j \in \{1, \ldots, L\}$, the finite set of all VPs in the MVP omnidirectional content.

Since the MVP omnidirectional streaming applications are still in their early stages of development, the interfaces for VPs switching that will become the most popular are not known yet. We present hereafter several options. We denote by $\mathcal{N}_{j,R} \subseteq \mathcal{V}$ the set of VPs accessible from a given viewport (v_j, R).

Teleportation without restriction Users can switch to any VP without restriction, formally $\mathcal{N}_{j,R} = \mathcal{V}$. It means the 6DoF VR application authorizes *teleportation*, without regards to the physics nor the visibility of the VP.

Step by step moves Users can only switch to neighbor VPs. Formally, $\mathcal{N}_{j,R}$ contains only the neighbors of v_j that are at most at distance d from v_j: $\mathcal{N}_{j,R} := \{v_k \in \mathcal{V} : 0 < \|v_k - v_j\| \leqslant d\}$. This option respects the law of physics since it binds the user to the most immediate moves in the Euclidean space. The user can decide to switch to a VP that is not in her current viewport: the use can move backward or sideways.

Teleportation within the viewport Users can switch to any VP in their current viewport. Teleportation is still possible, but users can only switch to visible VPs. Formally, $\mathcal{N}_{j,R}$ contains all VPs v_k within a certain angular distance from $\vec{\imath'}$, where $\vec{\imath'}$ denotes the the user's viewing direction (as introduced in Section 3.1).

Step by step moves within the viewport $\mathcal{N}_{j,R}$ is not only restricted to the nearest VPs to v_j, but also to the VPs that are located in the current viewport. This implementation is the most natural since it restricts the user to moving as in the real world. Formally, VP v_k belongs to $\mathcal{N}_{j,R}$ if and only if the *switching direction* v_k, is within a given angular distance from the user's viewing direction $\vec{\imath'}$ and if the distance between v_k and v_j is smaller than a given value d.

The temporal aspects of the switch are discussed in Section 4.

3.3 Multi-Viewpoint 360° Encoding Options

We consider the case of a content provider, which implements dynamic adaptive streaming technologies, such as MPEG DASH, to stream omnidirectional MVP video content to its clients. We do not consider any storage limitation at server side, *i.e.*, the server has unlimited storage capacity. The adaptation logic is implemented at client side, i.e., the client selects the video segments to request, since most of the adaptive streaming technologies implemented nowadays use this solution. The content provider has to choose how to encode the MVP 360° data.

We consider a scenario where the representation bit-rate varies but the resolution remains constant. The choice of the encoding solution results in different adaptation sets stored on the streaming server. In this paper, we consider two encoding options that result in corresponding adaptation sets (Figure 3):

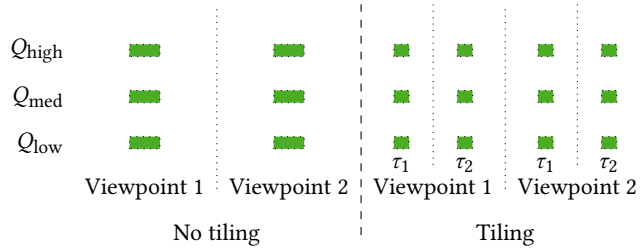


Figure 3: Adaptation Sets for a scenario without tile and an other with two tiles. In green, directly decodable base layers.

No Tiling The simplest encoding option to create the adaptation set is to encode each VP as one independent spatial entity, at multiple bit-rates. The adaptation set, as illustrated in Figure 3, contains one representation per bitrate (i.e., quality level) and per VP. Each representation can be downloaded independently of the others. To select the representation to download, the client needs a prediction algorithm for both available bandwidth and VP switching. When decoding the stream, the client uses only one representation per VP. All other downloaded representations are dropped and the bandwidth used to download them is wasted.

Tiling Each VP can be spatially divided into non-overlapping, independently decodable portions, i.e. *tiles*. The tile-based adaptive streaming optimization that has been proposed for mono-view omnidirectional video streaming can be extended to the MVP case. Each video corresponding to a VP is split into the same number $\mathcal{T}$ of tiles, according to the same tilling pattern. The adaptation set is then composed of $\mathcal{T}$ representations per VP and per bitrate (i.e., quality level), as illustrated in Figure 3 with $\mathcal{T} = 2$. Each tile can be downloaded and decoded independently of the others. Only one quality can be used by the decoder for a given tile and VP. Downloading tiles at different position in a given VP with different quality level allows the decoder to generate an output omnidirectional video with variable quality within the 360° frame. If the user watches a VP v_0, all downloaded representations of other VPs are wasted.

More complex encoding scenarios, exploiting the inter-view redundancy by using scalable video coding, have been successfully used to create adaptation sets for adaptive streaming of perspective videos [17, 20, 33], even in the case of multi-view perspective videos, as discussed in Section 2. While such scalable encoding solutions could be suitable for the MVP 360° scenario, eventually in combination with tiling, these options are out of the scope of this paper: we only consider encoding scenarios that do not use scalability.

4 CLIENT SIDE

The goal of the client is to download video segments so that it can display with a high quality the viewport requested by the user in the right VP. The client has a limited downloading capacity for each video segment. We denote by B_i the average available downloading bandwidth during video segment i.

4.1 Switching Decision and Timing

We first focus on VP switching, which is a key novel feature of the MVP 6DoF video streaming system. We suppose a user can at most switch VP once during a segment. We consider that any representation (whether it is the full omnidirectional video or a tile) has frequent Random Access Points (RAPs), *i.e.*, frames without any dependency to any prior frame. To display a given video frame, a client should have downloaded and decoded all the frames since the last RAP before the said frame. For the sake of simplicity and in conformance with the recommendations for implementation of dynamic adaptive streaming systems, we suppose that each segment in the adaptation set starts with a RAP and contains no other RAP.

Let us consider a user switching at instant t from a VP v_j to another VP $v_k \in \mathcal{N}_j$. To enable an *immediate* switch, the MVP omnidirectional streaming system should both (i) have in buffer the frames of the omnidirectional video of the VP v_k, and (ii) have decoded all the frames of VP v_k since the latest RAP. In practice, a standard user device does not concurrently decode multiple videos (multiple VPs of the same scene) due to memory and computing resource consumption. We thus consider two more realistic cases, whether the translation movement command results in a VP switch to the immediately following RAP of VP v_k or to the next one. We say that the segment identifier that is displayed at time t is the segment i. We denote by t_r this starting time of segment $i+1$.

- If the user already downloads the segment $i+1$ of VP v_k at instant t or if the user has still time to request the VP v_k so that the segment $i+1$ of v_k will arrive before t_r, then the switching can be done at t_r.
- If the user has not downloaded the segment $i+1$ of v_k and it has no time to download it until t_r deadline, then the switch cannot be done at the next RAP.

In the graphical interface, typically in Google Earth VR, the implementation of the translation movement includes a transition effect based on a "tunneling" animation, so that the switch appears as if it was instantaneous: upon the movement command, a blurred animation of the content (basically simulating motion) is shown until the time at which the display of the new VP can be done.

4.2 Download Decision and Scheduling

To fuel the video decoder buffer in input, we propose a serialized iterative request process with a unique buffer, as illustrated in Figure 4. The client issues a request for one representation at one segment in one VP. We suppose that once a segment is requested, the download cannot be canceled and has to be completed. Once the download is complete, the downloaded segment is stored in the buffer, and then the client can issue another request for another representation.

At the beginning of a video segment i, the client selects, among the video representations that are buffered for this segment i, the representation to decode. We suppose that (i) the client can only select fully downloaded and fully decodable representations; (ii) the client cannot start decoding another representations afterwards during the course of the segment; (iii) for each tile, at most one representation can be selected for decoding ; and (iv) the client can only select representations of the same VP (*i.e.*, cannot decode tiles from different VPs in parallel) .

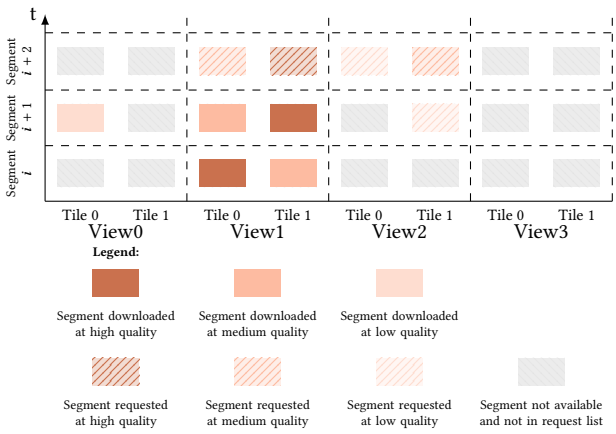


Figure 4: Snapshot of the buffer state at the beginning of segment i. Filled rectangles represent available segments, the darker the higher the quality. Striped rectangles represent not available segments. At this step, the client plans to request some of the not available segments.

4.3 Assessing QoE

Three objective metrics can be considered to evaluate the QoE felt by a user watching a MVP 360° video. Evaluating the correlation between those metrics and the subjective evaluation of the QoE is out of scope for this paper. Those metrics are:

Distortion The objective quality of the displayed viewports is related to the quality of the selected representations. For a given displayed representation, and for a given viewport during a segment, we extract the same viewport from the original full-quality omnidirectional video of this VP. This latter is the reference video. We can then run an objective video quality metric to compare the displayed viewport with the viewport from the original content. Video quality metrics include the Peak Signal to Noise Ratio (PSNR) (the average error of pixel intensities between the original and the displayed videos); the Multiscale - Structural Similarity (MS-SSIM) (the structural similarity between the two videos); and the Video Quality Metric (VQM) (the perceptual effects of video impairments between the two videos).

Stalls This event is when the client pauses the display of the content to wait for the next frames to be downloaded. A stall happens when no representation has been received on time. In the MVP scenario, a stall is likely to occur after unanticipated VP switch.

Viewpoint Switching Lag It is the duration between the time the user commands a translation move and the time the first viewport of the requested VP is displayed. The impact of this lag on the QoE is an open question. Even if not supported by any subjective testing, the shorter is the transition, the better is the feeling of immersion. It requires however the MVP omnidirectional system to anticipate the switch by downloading neighboring VPs, which may be not watched.

5 ALGORITHMS

We start this section by a discussion about the requirements of the algorithms that need to be implemented at the client side to enable good performance for 6DoF video streaming systems. Then, we study optimal algorithms, which correspond to some specific conditions. These algorithms aim to provide a theoretical framework for the study of MVP omnidirectional video systems and to highlight some radical implementation choices.

5.1 Toward Practical Algorithms

The system performance depends on the capacity of the client to predict (*i*) the network conditions in the next seconds, in particular the available bandwidth; (*ii*) the next head orientation rotations, which correspond to the interactive feature of traditional 3DoF video systems: yaw, pitch, and roll; (*iii*) the next translation movements, which are the three novel interaction features of 6DoF systems, here restricted to VP switch. We would like to emphasize again that the accuracy of the prediction algorithms on those three parameters is critical to the success of the implementation. The ongoing efforts to develop prediction strategies, for both traditional adaptive streaming and single-camera omnidirectional videos can be reused here.

The prediction of the client can then be used when the client has to take a decision on which representation to download next. This decision contains up to four choices:

- **The segment**. The request can be for a representation that will be played in the immediate next segment (typically if a VP switch just happened or if the latest head rotations reveal that previous head rotation predictions were wrong), or can be for longer term segments to anticipate the movements and have the time to download high-quality representations.
- **The bit-rate**. The higher the bit-rate the lower is the distortion. However the client should also take into consideration that requesting a high-quality representation will monopolize the bandwidth for a long time, at the expense of the next requests. Furthermore the client has to request representation such that the downloading can be completed before the decoding and displaying time.
- **The VP**. The prediction of the next translation movements make that the client can decide to request representations in the current VP (if the client does not anticipate any move in the near future) or in another VP (if the client anticipates a move soon). The client can also request representations in other VP as a proactive strategy in case of unanticipated moves.
- **The tile** (if implemented). It is here the prediction of the next head orientation rotations that make the client request tiles in various locations of the frame in the given VP. As for the VP, the request depends on the estimated accuracy of the prediction.

Multiple parameters have to be considered for the implementation of efficient algorithms in practice. In this paper, we do not design practical algorithms in this regard. We focus instead on analyzing optimal algorithms for two extreme strategies: a proactive strategy where the client always proactively downloads all other VPs to anticipate possible VP switches, and, on the contrary, a reactive strategy where the client reacts only to movement commands. Our analysis provides some bounds, which will hopefully serve a comparison basis for future work on practical algorithms.

5.2 Optimal Decision with Perfect Predictions

We first consider an omniscient client, which is able to perfectly predict the future available bandwidth, the future head orientation rotations, and the future switching decisions. The goal is thus to schedule the representation requests so that every video that is displayed in the viewport (*i.e.*, every displayed tile in the right viewport) is downloaded on time at the highest possible quality. For the sake of simplicity, we will not consider the case of scalable coding. The model can easily be upgraded by introducing a decoding dependency constraint to include this case. We present the model in the context of the tiling encoding scenario because the non tiling scenario can be seen as a tiling scenario with only one tile per VP.

The objective is to maximize the QoE for the user, with respect to the three metrics described in Section 4.3: distortion, stalls, and switch lag. We combine these three metrics into one by applying two float weight parameters α and β so that the function to maximize is a traditional image distortion metric, plus α times the stall metrics, plus β times the switch lag. Subjective quality assessment campaigns will be necessary to set these weights based on the relative impact of those metrics on the subjective QoE.

The notation is as follows. We denote by $r_{i,q,v,\tau}$ the representation for tile τ of VP v with quality level q for the segment i. To model this decision problem, we introduce a set of binary variables $x_{i,q,v,\tau}$ such that:

$$x_{i,q,v,\tau} = \begin{cases} 1, & \text{if the client selects the representation } r_{i,q,v,\tau} \\ & \text{to generate the viewports of segment } i \\ 0, & \text{otherwise} \end{cases}$$

We introduce the binary variables $y_{i,v}$ such that:

$$y_{i,v} = \begin{cases} 1, & \text{if VP } v \text{ is selected for display} \\ & \text{during segment } i \\ 0, & \text{otherwise} \end{cases}$$

The video distortion observed by the user for a segment i is computed as follows. We denote by $Q_{i,q,v,\tau}$ the average distortion observed by a user having its viewports totally inside the representation $r_{i,q,v,\tau}$ during the segment i. We suppose the distortion observed by a user having its viewports inside multiple tiles is equal to the weighted average of the average distortion of each tile representations, weighted by the ratio of the viewports surface within each tile. This is for instance the property of the Mean Square Error (MSE) when the distortion is evenly spatially distributed within each representation. We denote by $V_{i,\tau}$ the average ratio of the surface of the viewports of the user within the tile τ during the segment i. Then, the distortion observed by the user during the segment i is

$$Q_i = \sum_{v,\tau} V_{i,\tau} \cdot \Big(\sum_{q} (x_{i,q,v,\tau} \cdot Q_{i,q,v,\tau})\Big)$$

Other notations include $\ell_{i,v}$, which is a constant equal to 0 if the user wants to display the VP v during the video segment i, and 1 otherwise. The set of float variables $d_{i,j,q,v,\tau}$ is the downloading ratio of representation $r_{i,q,v,\tau}$ at the time of segment j. The integer variable s_j represents the duration (expressed in number of segment

duration) of a stall during the downloading segment j, and S_j is a binary variable equal to 1 if and only if s_j is greater than 0.

The optimal model can be formulated as follow:

$$\min_{\{x_{i,q,v,\tau}\}} \sum_i (Q_i + \beta \cdot l_{i,v} \cdot y_{i,v}) + \alpha \sum_j (s_j + S_j)$$

s.t.

$$\sum_{i,q,v,\tau} d_{i,j,q,v,\tau} \cdot b_{i,q,v,\tau} \leqslant (1 + s_j) B_j \qquad \forall j \quad (1a)$$

$$\sum_j d_{i,j,q,v,\tau} \leqslant 1 \qquad \forall i,q,v,\tau \quad (1b)$$

$$\sum_{j=i+1}^{N} d_{i,j,q,v,\tau} = 0 \qquad \forall i,q,v,\tau \quad (1c)$$

$$x_{i,q,v,\tau} = \sum_j d_{i,j,q,v,\tau} \qquad \forall i,q,v,\tau \quad (1d)$$

$$V_{i,\tau} \leqslant \sum_v \sum_{q=0}^{L} x_{i,q,v,\tau} \qquad \forall i,\tau \quad (1e)$$

$$\sum_v y_{i,v} = 1 \qquad \forall i \quad (1f)$$

$$x_{i,q,v,\tau} \leqslant y_{i,v} \qquad \forall i,q,v,\tau \quad (1g)$$

$$y_{i,v} \leqslant y_{i-1,v} + (1 - \ell_{i,v}) \qquad \forall i,v \quad (1h)$$

The set of equations (1) formally defines the optimization problem into an MILP problem. Equation (1a) defines the bandwidth constraint for each segment, including the extra bandwidth obtained when the client pause the video display. Equation (1b) limits each representation to be downloaded at most once. Equation (1c) forbids the download of a representation after its display deadline. Equation (1d) indicates that a representation can only be selected for decoding if it was fully downloaded, and forbids the optimal solution to download not displayed representations. Equation (1e) enforces the model to select one available representation when the tile is visible during the segment. The minimization problem implies that if this constraint is active, only one representation is selected. Equation (1f) guarantees that one and only one VP is displayed during a segment. Equation (1g) states that a representation of segment i can only be selected for display if it belongs to the displayed VP during segment i. Equation (1h) indicates that a VP can be selected for display either if the user requested to watch it or if it was displayed during the previous segment.

5.3 Optimal Proactive Strategy for Fast Switch

We consider now the case of a content provider that guarantees that the VP switching lag is minimal, *i.e.* whenever the user commands a translation move, the new VP is displayed at the next segment. This strategy can typically be implemented by a content provider that implements accurate bandwidth and head orientation prediction algorithms, but has no clue on the user behavior regarding translation movements.

The implementation of this strategy imposes to download, for every segment, at least one representation for the predicted tiles in every VP among all the possible switchable VPs. For instance, if the client is in VP v_j at segment i, then the requests for segment $i+1$ proactively include the tiles corresponding to the predicted head rotation R for *all* VPs in $\mathcal{N}_{j,R}$. To ensure a smooth transition, we also force the quality to be the same for all downloaded tiles of the same segment.

This strategy guarantees a minimal switching delay at the expense of the displayed video quality, wasted download resources, and stall if the bandwidth is too low. Indeed, at the start of a new video segment, representations for all VPs have been downloaded but only one VP is decoded and displayed. The price to pay for guaranteed interactivity is thus entire VP downloading waste. Note that the design choices of the content provider regarding the switching conditions (see Section 3.2) can have a significant impact on the performance.

To model this strategy and to allow fair comparison with the other tested strategies, we suppose a perfect bit-rate and head orientation prediction. The Equation (1d) is updated to additionally enforce the download of a representation if a representation with the same quality, at the same tile and segment was selected at any VP. We consider the worst case in Section 3.2: teleportation without restriction.

$$\sum_{v'} x_{i,q,v',\tau} = \sum_j d_{i,j,q,v,\tau} \quad \forall i,q,v,\tau \qquad (2d)$$

5.4 Optimal Reactive Strategy

We consider finally the opposite case of the previous strategy. The content provider now agrees that a VP switch can be postponed. It can typically be the case if the graphical animation for VP switching can be elegantly implemented. The prediction of bit-rate and head orientation is still perfect and, again, no prediction for translation movement is available. The client does not anticipate any VP switch, and thus implement a *reactive* strategy.

In this strategy, the client requests only representations in the current displayed VP. The strategy is said reactive because representations in the new VP are requested only when the user commands a translation moves. If the client has no time to download any representation until the start of the next segment, then the switch is postponed to the following segment. In this strategy, the waste of representation downloading is minimized, and thus the video quality is maximized.

To model this strategy, we insert the following constraints in the omniscient model:

$$d_{i,j,q,v,\tau} \leqslant y_{\max(j,0),v} + y_{\max(j+1,0),v} \forall i,j,q,v,\tau \qquad (3h)$$

If user switch to VP v at segment $j+1$:

$$\sum_{i,q,\tau} d_{i,j,q,v,\tau} \leqslant (1 - t_j + s_j) B_j \quad \forall v,j \qquad (3j)$$

Equation (3h) allows the client to download only representations that will be displayed during the next video segment or during the current video segment. Constraint (3j) indicates that if the user decides to switch VP t_j seconds after the beginning of the current downloading segment, the client cannot download segment for the new VP before the decision was made. We additionally add a constraint to enforce representations to be downloaded in display order (i.e. if a representation of video segment i is downloaded

during segment j then representations of segment $i-k$ cannot be downloaded during segment $j+k'$ with k, k' integers greater than 1).

6 EVALUATION

In this Section we evaluate the two extreme strategies introduced in Sections 5.3 and 5.4 and compare them to the optimal omniscient client defined in Section 5.2.

6.1 Test-bed

The evaluation test-bed is made of four main components.

- A four-minute long multi-view omnidirectional video, captured by three *Orah Live Spherical VR Camera 4i* 360° cameras. The cameras were aligned in a hallway, inside a building. Each camera showed in different offices with people working/moving inside. Each camera records a 4K (4096×2048) equirectangular video at 30 frame per second (fps). Cameras are positioned such that their local reference frames are aligned.
- An Android application to navigate into the MVP 360° video and record users' trajectories. The application is based on *Google Daydream*[3] framework. The neighboring VP are represented by small white sphere, as in Figure 1; users switch by using a controller. The video is streamed over Wi-Fi, with 2K resolution, at a constant quality level using DASH. The video is split into one-second long segments. The user can switch at any time but the switch is only effective at the beginning of the next segment. Around 60 times per seconds, the system logs a timestamp, the user's viewports orientation, and the current displayed VP. The switching decision times are also recorded in the log file.
- An offline optimization software, used to solve the three optimization problems introduced in Section 5. The input are the distortion and the bit-rate of each encoded video segment, the user trajectories inside the MVP video, and the average download bandwidth available during each download chunk. The outputs are the list of downloaded segments, the VP that was displayed to the user, and indication of the necessary stalls. The software is implemented in C++, and use the *IBM ILOG CPLEX Optimization Studio*. We released it open-source, and made it available on *Github*[4] to reproduce the results of this paper (cf. Appendix A).
- A Python3 script which combines *Gpac MP4Box* [23] and *ffmpeg*,[5] to generate the decodable bit-stream that the client would have obtained. The script then extracts, for each video frame, the viewport inside the generated video, with the orientation indicated by the user navigation trace, and in parallel inside the original video. This extraction is performed using an open-source software named *360Transformations*.[6]

We used the following procedure to encode the MVP 360° video. First we project the omnidirectional video of each VP onto a 3×2 compacted cube-map baseball as illustrated in Figure 2. For each tile set-up (no tile, 3×2, and 6×4 tiles), we used the open-source *Kvazaar* [19] software to encode each projected VP into three representations, encoded with an average bit-rate target: 5 Mbps, 8 Mbps or 16 Mbps. When tiles are used, we used the MCTS configuration of *Kvazaar*, and we enforced each tile to have the same dimensions, so that tiles are restricted to the faces of the cube-map projection. The resulting video were split into one-second DASH segments using *GPAC MP4Box*. We also used *MP4Box* to extract each tile of each video segment into a different bit-stream file. Table 1 summarizes the PSNR measured by *Kvazaar* during the encoding process, and the average bit-rate of each representation. The PSNR is measured in comparison to the original video, on the whole frame (even for the tiled videos), in the cube-map domain, on the luma component of the pixels. The bit-rate is computed on the DASH-ed files by adding the size of each file (including the initialisation files) and dividing by the video duration.

Finally, the objective function of the three optimization models uses the parameters α and β. The parameter α can be interpreted as a weight for the dissatisfaction on the user for every stall events; we set it to 100 times the maximal MSE in our adaptation set. The parameter β is the weight of the dissatisfaction for every second of lag; we set it to $\frac{\alpha}{10}$.

Videos	Quality 0		Quality 1		Quality 2	
	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)
no tile	46.25	5.01	47.12	8.01	48.45	16.02
3×2 tiles	46.22	5.02	47.10	8.02	48.44	16.02
6×4 tiles	46.19	5.08	47.06	8.08	48.42	16.08

Table 1: Bit-rate and distortion expressed in PSNR for the encoded video, average over all segments, over all tiles, and over all three cameras, for the three tiling scenarios and the three quality levels.

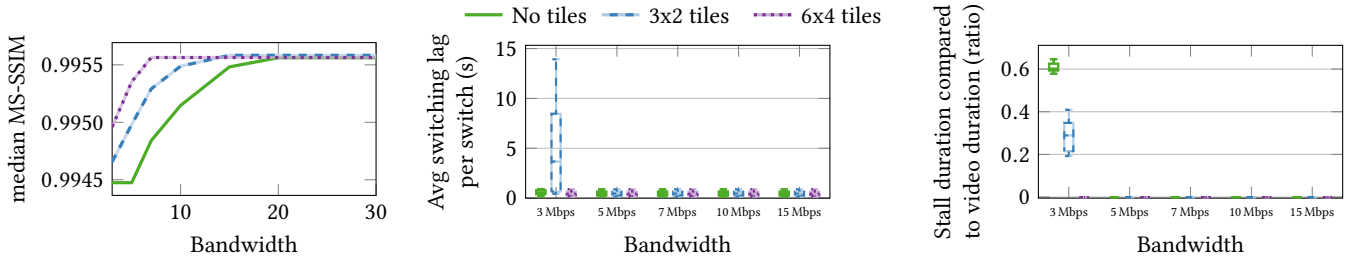
	User A	User B	User C	User D
Sex	F	F	F	F
Age	34	31	27	27
Nb. switch event	18	10	4	13
Ang. Vel. (Deg./s):				
global	28.81	20.88	28.44	20.00
2 s before a switch	8.48	3.42	3.24	7.53
2 s after a switch	59.57	46.88	54.25	51.21
between two switches	29.38	20.45	28.79	20.42

Table 2: Information about the users: sex, age, number of switching event, median angular velocity. The median angular velocity is ether computed globally, two seconds before a switch event, two seconds after a switch event or in the period between two switch events.

6.2 User Behavior in MVP 360° Video

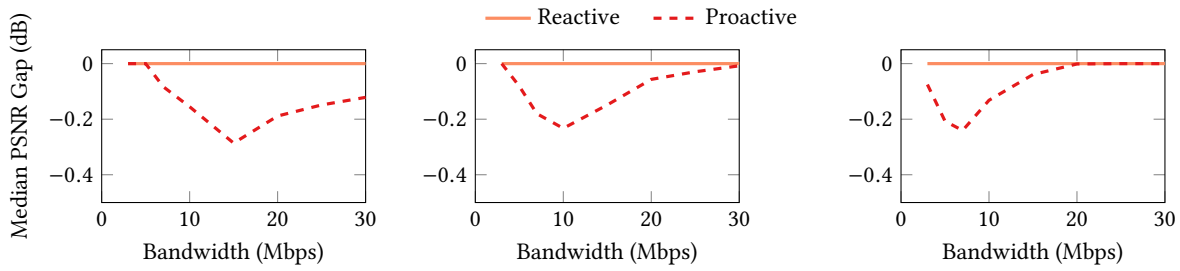
Table 2 shows some statistics about the navigation traces that we captured on four users (all female), whose age ranges from 27 to 34. All users had already watched single-viewpoint 360° videos. They switched to new VPs in average 11 times during the four-minute

[3] https://developers.google.com/vr/
[4] https://github.com/xmar/MultiViewpoint360_MMSys18
[5] https://www.ffmpeg.org/
[6] https://github.com/xmar/360Transformations/tree/master/transformation



(a) Median MS-SSIM within the displayed viewports (b) Box plot switching lag (c) Box plot stall ratio

Figure 5: QoE metrics for the omniscient scenario



(a) No tiles (b) 3x2 tiles (c) 6x4 tiles

Figure 6: Median of the gap between the displayed viewports PSNR of reactive or guaranteed scenario with the omniscient scenario.

long MVP video but the variance is here significant (more than four times more switches between user A and user D).

We extracted the median angular velocity from head movements, and we identified a *switch preparation time* and a *discovery time* respectively before and after switching events. Indeed, the median angular velocity is significantly lower than the median velocity two seconds before a switching event (around $20\,^\circ\,s^{-1}$ lower), while it is significantly higher two seconds after a switching event (around $30\,^\circ\,s^{-1}$ higher). Should these first observations be confirmed by more measurements, the prediction of VP switching events could become an easy process in standard behavior.

6.3 Results: Optimal with Perfect Prediction

We first analyse the choices of the omniscient optimal client, illustrated in Figure 5. This theoretical client uses its perfect knowledge of the future head orientation of the client to minimize the objective function. The objective function is split into three sub-objectives: minimizing the distortion in the displayed viewports, minimizing the time required to switch to a new VP, and minimizing the duration the video display was paused by the client. Our results show the interplay between these sub-objectives, with respect to our weight parameters α and β.

Figure 5a represents the median MS-SSIM [39] inside the video displayed in the viewports for various available bandwidth. The MS-SSIM objective metric compares image by image the structural similarity between the viewports displayed by the client and the same viewports extracted from the original video. It returns a value between 0 and 1. We observe that the distortion on the user viewports is reduced when the available download bandwidth increases. Moreover, the encoding with tiles allows the client to select only the displayed tiles in high quality, which results, for a given download bandwidth budget, in a viewport video with less distortions.

Figure 5b shows the average duration of a switching lag per requested VP switching, for different average available bandwidth budget, and different number of tiles. The results are represented inside a box plot (from bottom to top, the horizontal lines represent respectively the minimum, the 25^{th}, the median, the 75^{th}, and the maximum value of the average switching lags for a given scenario). When the budget is higher or equal to 5 Mbps, the client does not decide to delay the switch.[7] When the bandwidth is less than 5 Mbps the client does not have enough bandwidth to download a full VP at the lower quality level (encoded at 5 Mbps) and it has sometimes to delay the switch to allow the display of viewports with better quality.

Figure 5c depicts the cumulated stall duration relative to the duration of the video. The results are also represented in box plots, which are defined as the box plots in Figure 5b. When the bandwidth is higher than 5 Mbps (at least the bit-rate of the video with lowest quality), the client decides to never pause the video. However, when the bandwidth is not enough, the client decides to pause the video to get extra time to download better quality segments.

We highlight the complex interplay between the three options in case of bandwidth shortage: reduce visual quality by switching

[7] The small variations in the switching delay come from the fact the users switches at any time in the course of a segment.

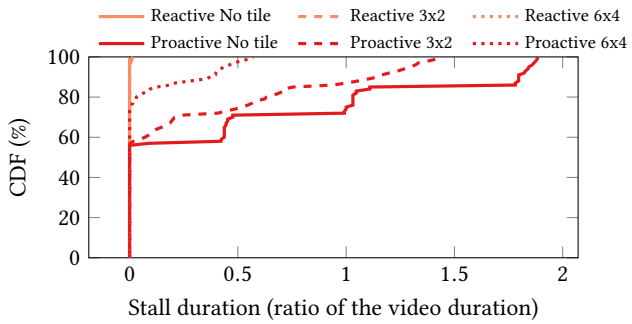


Figure 7: CDF stall duration compared to video duration, aggregated for bandwidth between 5 Mbps to 20 Mbps

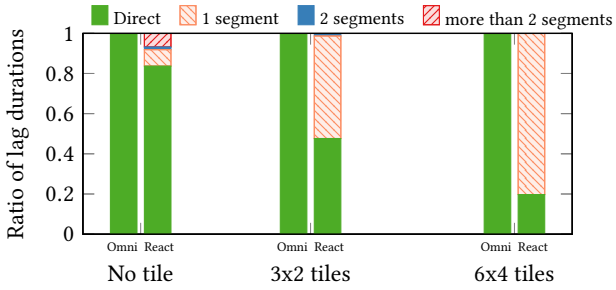


Figure 8: Ratios of lag durations in segment duration unit, aggregated for bandwidth between 5 Mbps to 20 Mbps, for the different scenarios and for different number of tiles

to lower quality representations, or pause, or delay a switch, or combinations of them. Here, the omniscient optimal client picks in every of these behaviors to optimize the objective function with regards to the set weight parameters. This complex behavior calls for a campaign of subjective tests to model the parameters that reflect the interplay with better accuracy.

6.4 Results: Proactive and Reactive Strategies

We now compare the performance and the behavior of both client strategies against the optimal performance of the omniscient client. These two strategies are extremum of the possible downloading strategies that the client may implement. The *proactive* strategy does not accept any compromise on the VP switching. The excessive anticipative VP downloading generates bandwidth shortage, which can only be mitigated by stalls and distortion. On the contrary, the *reactive* strategy tolerates switch lags, with expected gains regarding video quality and stalls.

Figure 6 represents the median of the difference between the PSNR in the viewports for both strategies and the PSNR in the viewports of the omniscient client. The *reactive* client manages to obtain viewport videos with distortion close to the optimal, although the viewport distortion of the *proactive* client is higher. Note that the proactive strategy compensates the bandwidth shortage in different ways. For very high shortage (3 Mbps bandwidth), the client prefers to pause the video long enough to get a good quality video. However, the quality increases less quickly with the increase of the bandwidth than for the omniscient client. We also observe the advantage of using tiles: the client can more efficiently use the available bandwidth to get high-quality images in the user viewing direction. This effect is amplified by our assumption of perfect head orientation prediction.

Figure 7 depicts the cumulative density function (CDF) of the stall duration, relatively to the video duration, for both strategies. The CDF is generated using the stall duration for each user, aggregated for available bandwidth ranging from 5 Mbps to 20 Mbps. The proactive client has to pause the video display for a duration at least equal to the video duration in 25 % of the cases when no tile are used, and in 20 % of the cases when 3×2 tiles are used. There is at least one stall of 250 ms in 20 % of the case for the 6×4 tiles scenario. This long stalls are the price to pay for downloading the tiles in all the VPs (which guarantees minimum-lag VP switch). The client requires around three times more download bandwidth than the two other clients to download a given displayed tile at a given quality. On the contrary, the reactive client never pauses the video for the 6×4 and 3×2 tiles scenarios and pause in only 5 % of the cases for the no tile scenario. Increasing the number of tiles decreases the needs for both strategies to pause the video.

Figure 8 represents the distribution of the delay between the switch command and the actual display of the new VP. The delay is measured in terms of number of segments. The results are aggregated for the download bandwidth ranging from 5 Mbps to 20 Mbps. We do not represent the *proactive* strategy since the switch is by definition always performed immediately after the command. The optimal omniscient does not need any delay to switch VP for this bandwidth, which means that a perfect prediction of VP switches enables high-quality interactive QoE. On the contrary, the *reactive* strategy needs to compensate the unanticipated switch events by delaying VP switch. Depending on the setting of the tile encoder, the strategy differs. When no tile is implemented, the strategy manages to switch immediately in 84 % of cases, but it compensates by a few more stalls and, more importantly, switch delayed by two or more segments in 9 % of cases. The flexibility offered by the tiling encoding enables the reactive strategy to implement more consistent switches in the 6×4 and 3×2 tiles scenarios. Then, no switch is delayed by more than one segment.

7 CONCLUSION

In this paper, we discuss the new challenges that are brought by 6DoF VR applications. We focus on a restricted version of 6DoF applications where the scene is simultaneously captured by several synchronized omnidirectional cameras. This paper shows that such an application permits different implementations of VP switching. It also describes different options to encode the MVP videos into segments friendly to existing encoders and adaptive streaming technologies. The paper introduces the key trade-off the client has to consider when scheduling the video segment downloading to maximize the user experience. We identify the main objective metrics that correlate with the user's feeling of immersion and QoE. We design an optimization model, which bounds the achievable performance, and two client strategies: a reactive client and a proactive client with guaranteed fast VP

switch. A real MVP omnidirectional video is used, and real user navigation traces are exploited to evaluate the performance of these algorithms. Our main observations include that (*i*) tiling improves service performance and is thus a key technology for 6DoF VR applications, and (*ii*) proactive strategies based on anticipated systematic neighboring VP downloads represent an excessive price to pay. A compromise between the reactive strategy (which tends to delay switches) and the proactive strategy (which has to pause the videos to get a decent image quality) has to be found, but to date, the reactive strategy seems a better option.

Being a first step in the field of MVP 360° video systems, our paper reveals many open research questions, including (*i*) VP switching prediction (the early results show that it should be possible to exploit the change in user behavior before switching); (*ii*) subjective tests to better understand the interplay between the various options when unanticipated events happen; (*iii*) graphical transition effects during switch, which may allow for delayed VP switches by increasing image quality and reduce stalls; (*iv*) exploiting at the client side multiple decoders in parallel at the client to allow near frame delay VP switching; (*v*) novel encoding strategies based on multi-view correlations, to increase the number of virtual cameras, and thus to get closer to continuous 6DoF VR applications.

REFERENCES

[1] URL https://vr.google.com/earth/.

[2] N. Carlsson, D. Eager, V. Krishnamoorthi, and T. Polishchuk. Optimized adaptive streaming of multi-video stream bundles. *IEEE Trans. on Multimedia*, 19(7): 1637–1653, July 2017.

[3] C. Concolato, J. Le Feuvre, F. Denoual, F. Maze, N. Ouedraogo, and J. Taquet. Adaptive streaming of hevc tiled videos using mpeg-dash. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[4] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *Proc. of IEEE ICC*, 2016.

[5] X. Corbillon, F. De Simone, and G. Simon. 360-degree video head movement dataset. In *Proc. of ACM Multimedia Sys. (MMSys)*. ACM, 2017.

[6] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski. Optimal Set of 360-Degree Videos for Viewport-Adaptive Streaming. In *Proc. of ACM Multimedia (MM)*, 2017.

[7] L. D'Acunto, J. van den Berg, E. Thomas, and O. Niamut. Using MPEG DASH SRD for zoomable and navigable video. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2016.

[8] M. Domański, O. Stankiewicz, K. Wegner, and T. Grajek. Immersive visual media-MPEG-I: 360 video, virtual navigation and beyond. In *Proc. of IEEE Int. Conf. on Sys., Signals and Image Proc. (IWSSIP)*, 2017.

[9] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu. Fixation prediction for 360° video streaming in head-mounted virtual reality. In *Proc. of ACM NOSSDAV*, 2017.

[10] V. Gaddam, H. Ngo, R. Langseth, C. Griwodz, D. Johansen, and P. Halvorsen. Tiling of Panorama Video for Interactive Virtual Cameras: Overheads and Potential Bandwidth Requirement Reduction. In *Picture Coding Symposium (PCS)*, 2015.

[11] M. Graf, C. Timmerer, and C. Mueller. Towards bandwidth efficient adaptive streaming of omnidirectional video over http: Design, implementation, and evaluation. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2017.

[12] A. Hamza and M. Hefeeda. A dash-based free viewpoint video streaming system. In *Proc. of ACM NOSSDAV*, 2014.

[13] A. Hamza and M. Hefeeda. Adaptive streaming of interactive free viewpoint videos to heterogeneous clients. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2016.

[14] M. Hosseini and V. Swaminathan. Adaptive 360 VR video streaming based on MPEG-DASH SRD. In *Proc. of IEEE ISM*, pages 407–408, 2016.

[15] M. Hosseini and V. Swaminathan. Adaptive 360 vr video streaming: Divide and conquer. In *International Symposium on Multimedia (ISM)*, pages 107–110. IEEE, 2016.

[16] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-DOF VR videos with a single 360-camera. In *IEEE Virtual Reality (VR)*, 2017.

[17] S. Ibrahim, A. H. Zahran, and M. H. Ismail. Svc-dash-m: Scalable video coding dynamic adaptive streaming over http using multiple connections. In *2014 21st International Conference on Telecommunications (ICT)*, pages 400–404, May 2014. doi: 10.1109/ICT.2014.6845147.

[18] ISO/IEC 23000-20. Omnidirectional media application format (omaf) committe draft. MPEG, November 2017. ISO/IEC JTC1/SC29/W11.

[19] A. Koivula, M. Viitanen, A. Lemmetti, J. Vanne, and T. D. Hämäläinen. Performance evaluation of Kvazaar HEVC intra encoder on Xeon Phi many-core processor. In *IEEE Global Conf. on Signal and Info. Proc. (GlobalSIP)*, 2015.

[20] C. Kreuzberger, D. Posch, and H. Hellwagner. A scalable video coding dataset and toolchain for dynamic adaptive streaming over http. In *Proc. of ACM Multimedia Systems (MMSys)*, pages 213–218. ACM, 2015.

[21] E. Kuzyakov. End-to-end optimizations for dynamic streaming. Blogpost, February 2017. https://code.facebook.com/posts/637561796428084.

[22] E. Kuzyakov and D. Pio. Next-generation video encoding techniques for 360 video and vr. Blogpost, January 2016. https://code.facebook.com/posts/1126354007399553.

[23] J. Le Feuvre, C. Concolato, and J.-C. Moissinac. GPAC: open source multimedia framework. In *Proc. ACM Multimedia Conf. (MM)*, 2007.

[24] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733, 2014.

[25] C. Liu, I. Bouazizi, and M. Gabbouj. Rate Adaptation for Adaptive HTTP Streaming. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2011.

[26] W. Lo, C. Fan, J. Lee, C. Huang, K. Chen, and C. Hsu. 360° video viewing dataset in head-mounted virtual reality. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2017.

[27] F. Michelinakis, N. Bui, G. Fioravantti, J. Widmer, F. Kaup, and D. Hausheer. Lightweight mobile bandwidth availability measurement. In *Proc. of IFIP Networking*, 2015.

[28] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou. An Overview of Tiles in HEVC. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):969–977, dec 2013.

[29] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim. MPEG DASH SRD: spatial relationship description. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2016.

[30] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proc. of ACM Multimedia (MM)*, 2017.

[31] F. Quan, B. Han, L. Ji, and V. Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *ACM SIGCOMM AllThingsCellular*, 2016.

[32] S. Rossi and L. Toni. Navigation-aware adaptive streaming strategies for omnidirectional video. In *Proc. of IEEE MMSP*, 2017.

[33] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louédec. Efficient http-based streaming using scalable video coding. *Image Commun.*, 27(4):329–342, Apr. 2012. ISSN 0923-5965.

[34] I. Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE MultiMedia*, 18(4):62–67, 2011.

[35] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *Proc. of IEEE ISM*, pages 583–586, 2016.

[36] T. Stockhammer. Dynamic adaptive streaming over http –: Standards and design principles. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2011.

[37] T. Su, A. Sobhani, A. Yassine, S. Shirmohammadi, and A. Javadtalab. A DASH-based HEVC multi-view video streaming system. *Journal of Real-Time Image Processing*, 12(2):329–342, Aug 2016.

[38] L. Toni and P. Frossard. Optimal representations for adaptive streaming in interactive multiview video systems. *IEEE Trans. on Multimedia*, 19(12):2775–2787, Dec 2017.

[39] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Int. Conf. on Signals, Systems and Computers*, 2003.

[40] C. Wu, Z. Tan, Z. Wang, and S. Yang. A dataset for exploring user behaviors in VR spherical video streaming. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2017.

[41] J. Xiao, M. M. Hannuksela, T. Tillo, and M. Gabbouj. A paradigm for dynamic adaptive streaming over http for multi-view video. In Y.-S. Ho, J. Sang, Y. M. Ro, J. Kim, and F. Wu, editors, *Advances in Multimedia Information Processing – PCM 2015*, pages 410–418, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24078-7.

[42] R. G. Youvalari, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Efficient coding of 360-degree pseudo-cylindrical panoramic video for virtual reality applications. In *Proc. of IEEE ISM*, pages 525–528, 2016.

[43] D. Yun and K. Chung. DASH-based multi-view video streaming system. *IEEE Trans. on Circuits and Systems for Video Technology*, PP(99):1–1, 2017.

[44] W. Zhang, S. Ye, B. Li, H. Zhao, and Q. Zheng. A priority-based adaptive scheme for multi-view live streaming over http. *Computer Communications*, 85:89 – 97, 2016. ISSN 0140-3664.

[45] M. Zhao, X. Gong, J. Liang, J. Guo, W. Wang, X. Que, and S. Cheng. A cloud-assisted DASH-based scalable interactive multiview video streaming framework. In *Picture Coding Symposium (PCS)*, 2015.

[46] C. Zhou, Z. Li, and Y. Liu. A measurement study of occulus 360 degree video streaming. In *Proc. of ACM Multimedia Sys. (MMSys)*, 2017.

A OPEN SOFTWARE

Alongside this paper, we release a part of the software and of the dataset used to generate the results presented in Section 6. In this Annexe, we describe briefly this software and where you can find it.

A.1 Description

The open software is available on *Github* at the following web address: https://github.com/xmar/MultiViewpoint360_MMSys18. Readme files and docker containers are available to help the reader to run the software.

The software is split into two main pieces:

MILP_Multiview contains the C++ implementation of the MILP presented in Section 5, using the *IBM ILOG CPLEX Optimization Studio*.

reconstruct contains a *Python3* [8] script that read the output results from the MILP to construct the bitstream as the client would have received it, and use the user head movement records to extract the viewport in the original videos and in the reconstruct bitstream to compute distortion metrics.

Docker[9] containers are available to run the software, and bash script are available to start the docker containers with the right shared resources. The docker images are available on *docker hub*[10] with the tag *mmsys18*, but the reader can build the images from scratch using the *Dockerfile*s available in the repository. The docker images were compiled and tested using docker version *18.03.0-ce* on an Archlinux (4.15.12-1-ARCH x86_64) machine.

The *reconstruct* script requires docker to run the 360Transformations[11] software.

The video dataset is downloaded when running the bash script inside the *reconstruct* folder. Only the 34 s of the video are currently available.

The raw head movement and viewpoint switching dataset is available in the folder *rawNavigationTrace*.

A.2 Docker Installation

It is not mandatory to use docker to run the different pieces of software but we only provide instructions to use the docker containers.

If docker is not installed on your machine, please follow the instructions to install the docker Community Edition available on docker official website[12]. Docker website provides installation instructions for all major operating systems.

Do not forget to start the docker server before trying to run the containers.

A.3 License

The software is released under the *MIT* license[13].

A.4 Usage Examples

In this Section we indicates how to run the two pieces of software with docker. It may take a few hours to run each pieces of software. If you want to build the docker images yourself, please read the README files in the github repository. In the following we suppose docker is already installed on running on your machine.

To run the MILP program, open a terminal inside the folder named *MILP_Multiview*, and run the bash script named *./runDockerContainer.sh*.

To run the viewport extraction program, open a terminal inside the folder named *reconstruct*, run the bash script *./buildDockerContainer.sh* to build the last layer of the docker images and then run the bash script *./runDockerContainer.sh*.

[8] https://docs.python.org/3/
[9] https://www.docker.com/
[10] https://hub.docker.com
[11] https://github.com/xmar/360Transformations/tree/master/transformation
[12] https://docs.docker.com/install/
[13] https://opensource.org/licenses/mit-license.php